



(19)

Europäisches Patentamt

European Patent Office

Office européen des brevets



(11)

EP 0 813 139 A2

(12)

EUROPEAN PATENT APPLICATION

(43) Date of publication:
17.12.1997 Bulletin 1997/51(51) Int. Cl.⁶: G06F 3/033

(21) Application number: 97304137.9

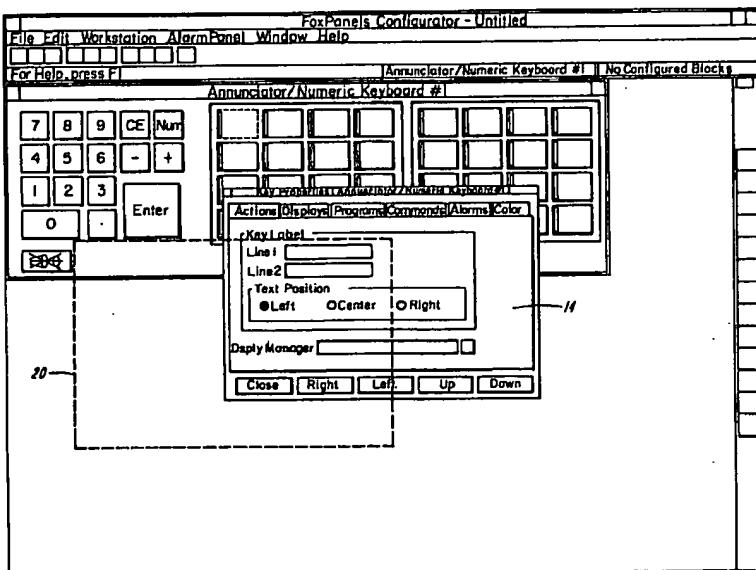
(22) Date of filing: 13.06.1997

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC
NL PT SE(30) Priority: 14.06.1996 US 19782 P
31.01.1997 US 792971(71) Applicant: THE FOXBORO COMPANY
Foxboro, MA 02035 (US)(72) Inventor: Calder, Dale E.
Mansfield, Massachusetts 02048 (US)(74) Representative:
Downing, Michael Philip et al
Fry Heath & Spence,
The Old College,
53 High Street
Horley, Surrey RH6 7BN (GB)

(54) Dynamic property sheet system

(57) Methods and apparatus for the concurrent display of two or more property pages of a Graphical User Interface (GUI) are disclosed. A user can select a property page displayed by a first property sheet system and create a second property sheet system that displays the selected property page at second, user selected display position. Alternatively, the user can place, or dock, the selected property page into an existing property sheet system for inclusion therein and display thereby. The first property sheet system displays, at a first display

position, any of the property pages remaining therein. Typically, the selected property page is removed from the first property sheet system and the first property sheet system is destroyed if it contains no other property pages. The invention allows a user of a GUI to concurrently display and interact with multiple property pages of multiple property sheet systems for optimizing the exchange of information with the GUI.

FIG.
3

EP 0 813 139 A2

Description

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

Field of the Invention

10 The present invention relates to graphical user interfaces, and particularly to property sheet systems having property pages viewable on a graphical user interface.

Background

15 A graphical user interface (GUI) is an indispensable feature of most modern computer systems. A GUI allows a user to interact intuitively with an application program running on the computer by allowing the user to manipulate pictorial representations of objects. Many of the display components of a typical GUI, such as title bars, menu bars, icons, multiple windows, scroll bars, pop up menus and dialog boxes, are now standard features. A modern GUI represents a major improvement over the text-based entry systems of just a few years ago.

20 Typical dialog boxes of a GUI system allow a user to exchange information with an application program. Some dialog boxes simply require a user to confirm an operation before it is executed by the computer; other dialog boxes may ask for user selected input, such as choice of a disk drive, directory, file name, file type, network path, etc. Dialog boxes typically comprise components such as text boxes, list boxes, option buttons, and command buttons.

25 If a large amount of related information is to be exchanged between a user and the computer a GUI may include a special type of dialog box known as a property sheet system. A property sheet system, for example, can have a display that includes several tabs which represent different dialog boxes, known as a property pages. The user can select a tab to display the particular property page associated with that tab. Once selected, the user can enter information, confirm an operation, or make other appropriate option choices.

30 Available property sheet systems can be frustrating to use. For example, to configure optimally an application program, a user often exchanges information with the program by using several modeless property pages. Often the configuration options that result in the desired performance of the application are not immediately obvious and are arrived at by some trial and error. Unfortunately, because the property sheet system displays only one property page at a time on a monitor, a user usually selects appropriate tabs to switch between multiple property pages, e.g. property pages are relegated to the background Furthermore, the user is typically required to remember the particular settings on the 35 nondisplayed property pages. This procedure of continually switching between property pages to configure a system is overly laborious and time consuming.

Accordingly, an object of the invention is to provide a property sheet system that allows a user to easily and efficiently exchange information with an application program.

40 A further object of the invention is to provide a property sheet system that reduces the need of a user to choose continually between displayed and nondisplayed property pages when the exchange of information requires the use of more than one property page.

Other general and specific objects of the present invention will be apparent and evident from the accompanying drawings and the following description.

45 Summary of the Invention

The present invention relates to, broadly, the creation of a second property sheet system from a first property sheet system. The first property sheet system includes one or more property pages, which are displayable on a display monitor at a first display position. The first property sheet system is often part of an existing application program. For example, a word processor program on which this application is being typed has a property sheet system for setting up printers. This property sheet system includes "tabbed" property pages entitled "setup", "paper", "destination", "print quality", and "fonts". According to the invention, a property page is selected in response to an input signal from a user-operated pointing and selecting device, such as a mouse. In response to a second signal generated by the user, typically via the mouse, the selected property page is included in, and displayed by, a second property sheet system. The 50 user may thus display and interact with two property pages simultaneously: the selected property page, which is displayed by the second property sheet system, and any one of the property pages remaining in the first property sheet system, which is displayed by the first property sheet system.

One technique by which a user may select a property sheet is by "tearing," that is, selecting a tab representing a property page of the first property sheet system. The property page "torn" or selected from the first property sheet sys-

tem is displayed at a second location, determined by the user, on the display screen. Typically, a user employs "drag and drop" operation to select a property page, create a second property sheet system, and to display the torn property page at a user-selected display location.

As noted, creating more than one property sheet system allows a user to view and exchange data with an application program using two or more property pages that are viewed simultaneously. The user need not continually switch between displayed and non displayed property pages, nor need he or she commit to memory the contents of one or more property pages. Note that as used herein, the term property page or sheet is intended to include a modal or modeless secondary window that displays user-accessible properties of an object. A property page is said to be included in a property sheet system and can be displayed in a variety of formats to best allow a user to exchange information with a computer or with an application program running on a computer.

Additionally, property pages can be exchanged between property sheet systems, whether these systems are created according to the invention or exist as part of a separate application program, such as a word processor program. A property page "torn" from a property sheet system is typically displayed as a geometric object or other symbol during the "drag" portion of the "drag and drop" operation. A user can place the selected, or "torn," property page in an existing property sheet system for inclusion therein and display thereby, rather than in a new property sheet system created to include the selected page. Typically, the user places the symbol representing the "torn" page in close proximity to a property page displayed by the property sheet system into which it is desired to drop the "torn" page. A docking input signal, such as release of a mouse button, causes the selected property page to "drop", or "dock" into the chosen property sheet system. Once docked in the chosen system, the property page may be used by the user just as when it was in the previous property sheet system.

According to one aspect, the invention includes a processor; a memory element; a display monitor; a user input device; a selection element for selecting a property page of a first property sheet system responsive to a signal generated by a user input device, such as, for example, mouse, trackball, or keyboard; and a second program element for creating a second property sheet system for including the selected property page and for displaying the selected property page at a second display position on the display monitor.

According to another aspect, the invention includes a removal element for removing from the first property sheet system the property page selected by the selection element such that the selected property page is not displayable by the first property sheet system.

In yet another aspect, the invention includes a destruction element for destroying a property sheet system when system does not include any property pages, such as when the last page in the system has been removed by the removal element.

According to another aspect of the invention the invention further includes a docking element, for, in response to a signal generated by a typical user input device, placing a property page selected by the selection means into a selected property sheet system for inclusion therein and display thereby.

In yet another aspect, the invention incorporates a positioning program element, responsive to input signal generated by the user input device, for selecting the second display position at which the second property sheet system displays property sheets included therein.

In a different variation, the positioning element varies the display position of a symbol in response to a user input. The second program element, responsive to the user input, displays the selected property page at the second position. The positioning element also allows for placing the symbol in substantial registration with a displayed property page of a property sheet system such that the docking element, responsive to a user input signal, such as release of a mouse button, places the selected property page into the property sheet system with which the symbol was in substantial registration. The symbol can be the dotted outline of a box, and the box can be the approximate size of a displayed property page.

To aid the user in docking a selected property page into an existing property sheet system, the invention can include an element for displaying on the display monitor an icon having a second display state for indicating to a user when the symbol or geometric object is in substantial registration with a property page of a property sheet system. For example, the icon second display state can comprise a display of a two puzzle pieces that are mated, and the icon can have a first display state comprising two puzzle pieces that are not mated, for indicating when a symbol or geometric figure is not properly registered, i.e., the selected property page will not be docked by the docking means into an existing property sheet system.

According to another feature of the invention, in addition to a positioning element responsive to user input for varying a position of symbol displayed on the display monitor, a a proximity program element is included. The proximity element determines a parameter responsive to the distance on the display monitor between the symbol and the position on the display monitor of a displayed property page of a property sheet system. When the parameter is less than a first amount, the docking means, responsive to a first user input signal generated by the user input device, places the property page selected by the selection element into the property sheet system for inclusion therein and display thereby. When the parameter varies from the selected value by a second selected amount, the second program means, responsive to a second input signal, places the property page in a newly created property sheet system and displays the

selected property page at a position on the display monitor substantially determined by the display position of the symbol. For example, the displayed property page can be centered about the position of the symbol. The first and second input signals can be responsive to the release of a mouse button as part of the "drag and drop" procedure.

Of course, the invention pertains not only to a computer programmed in the manner described above to perform the transformation and operations described above, but to a machine readable data storage medium including the above-described elements. The invention is also understood to comprise methods for performing the above operations.

It will thus be seen that certain changes may be made in the above constructions without departing from the scope of the invention. It is intended that all matter contained in the above and following description or shown in the accompanying drawings be interpreted as illustrative and not in a limiting sense.

It is also to be understood that the following claims are to cover all generic and specific features of the invention described herein, and all statements of the scope of the invention which, as a matter of language, might be said to fall therebetween.

Brief Description of the Drawings

The foregoing and other objects, features and aspects of the invention will be apparent from the following description and apparent from the accompanying drawings, in which like reference characters refer to the same parts throughout the different views. The drawings illustrate principles of the invention and, although not to scale, shown relative dimensions.

FIGURE 1 illustrates a typical computer system according to the invention.

FIGURE 2 illustrates a typical property page of a property sheet system for configuring, for example, the keyboard display of an application program. A user accesses each property page of the property sheet system by selecting the appropriate tab displayed at the top of the property page window display.

FIGURE 3 illustrates an intermediate display presented as part of the "drag and drop" procedure for creating a new property sheet system including a property page "torn" from the first property sheet system of FIGURE 2.

FIGURE 4 illustrates a second property sheet system including the property page "torn" from the first property sheet system illustrated in FIGURE 2 and which is displayed simultaneously therewith.

FIGURES 5A and 5B depict the selected display states of a dynamic icon for aiding a user in "docking" a property page removed from one of the property systems for placement within another sheet system.

FIGURE 6 is a flow chart diagram illustrating the steps involved in removing a property page from a property sheet system and placing it in a second property sheet system, as depicted in FIGURES 2-4.

FIGURE 7 is a schematic illustration block diagram of the software components utilized in accordance with the teachings of the present invention;

FIGURE 8 illustrates the object classes, and their relationships, in accordance with the teachings of the present invention.

FIGURES 9A-9C illustrate parent-child relationships of particular object classes depicted in FIGURE 8 and used in the software coding according to the teachings of the invention.

FIGURE 10 is a hierarchical schematic illustrating the derivation of key property pages from a property page base class according to the teachings of the present invention.

Description of Illustrated Embodiments

The removing or tearing of property pages from a property sheet system, in accordance with the teachings of the present invention, extends the user interface of standard Windows/Windows 95/Windows NT property pages to enable a user to change values on separate property pages simultaneously without changing visual context. The user can grab the active property page and pull/tear it away from the property sheet system and place it in an existing or newly created property sheet system. However, one of ordinary skill in the art, in accordance with the teachings herein, understands the invention to be applicable to GUI's in other operating environments, such as, for example, X-Windows, Motif, and IBM OS/2.

The practice of the invention includes a computer, such as the computer 2 illustrated in FIGURE 1. The computer 2 typically includes input-output hardware 3, a central processor 4, random access memory 5, disk storage 6, a keyboard 7, pointing device, such as a mouse 8 and a display monitor 10. The Input/output hardware 4 connects the computer 2 to, for example, a network.

FIGURE 2 illustrates a typical computer display screen display 12 of a property page 14 of a property sheet system.

The property page 14 in FIGURE 2 is used for configuring the keyboard display 16 of an application program that monitors and controls industrial processes. One such industrial process control system is the I/A SERIES® System available from The Foxboro Company, of Foxboro, Massachusetts, USA. However, an industrial process monitoring system is merely one example of the type of environment in which property pages are typically very useful; the present invention is useful in many computer application programs, such as word processors, spreadsheets, communications pack-

ages, web browsers, etc. However, in the particular embodiment illustrated here, the property pages are used to configure a keypad annunciator of an industrial process control system, such as the keypad annunciator 16 in FIGURE 2. Many of the keys of a keypad annunciator 16 are configurable, via property pages, such that a given individual key is associated with a particular alarm state of the industrial process control system; occurrence of that alarm state causes the key to change appearance. The key is also configurable such that selecting the key with the mouse can initiate selected corrective action or display a selected process graphic.

To "tear" a page from a property sheet system, the user positions a cursor over the body (i.e., generally anywhere other than an edit field or the tab representing the property page 14) of property page 14 and provides an input signal, typically by depressing a button on the mouse. During this tearing, or selection, process the user is presented with an outline of the property page while the mouse is pressed, and the outline can be positioned by "dragging" it anywhere on the display screen. FIGURE 3 illustrates the outline 20 of property page 14 that is in the process of being torn from its property sheet system. Once the desired position is found, releasing the mouse button signals the application to create a new property sheet dialog system and populate it with the torn-away property page. Typically, the original property page 14 is removed from its original container, e.g., the first property sheet system, and is only available in the second property sheet system. The user can view and change values using different property pages in one visual context, that is, to view them simultaneously positioned above the desktop 13. FIGURE 4 illustrates the simultaneous display of two property pages, the torn page 14, displayed in the display window 24 of the newly created property sheet system, and property page 22, now displayed in the original property sheet system display window 15, and corresponding to the tab 25 labeled "displays." The tab 18 labeled "actions" in FIGURE 2 is removed from property sheet display window 15 and now appears in display window 24 in the newly created property sheet system, as illustrated in FIGURE 4. The property page 14 and the property page 22 are now displayed concurrently, and the user can exchange data with either of them. Typically, as the user interacts with one of the displayed property pages, the other is updated, as necessary, to reflect the data exchange with the other. For example, if the display window 24 of the newly created property sheet system is active, that is, the user is exchanging data with the property page 14, the property page 22, displayed in the non-active window 15 of the original property sheet system, is updated, as necessary, to reflect any data input by the user to the property page 14 displayed in window 24.

In addition to being able to create a new property sheet system for displaying the selected property page, the user can place the selected property page into an existing property sheet system. The user attaches, or docks, a "torn" property page into an existing property sheet system by dragging the outline 20 over the top of a property page displayed by that system. Releasing the mouse button when the outline is so positioned docks the property page into the property sheet system. As discussed immediately above, a displayed property page in a non active window is updated, as necessary, to reflect any input by a user to a displayed property page in an active window. Using the docking procedure, the property page 14 in FIGURE 4 can be selected and placed back into the original property sheet system for re-display in display window 15.

A visual cue, such as a dynamic icon 28 in FIGURES 5A and 5B, is provided to assist in the tear-away/docking operation. During the dragging process, the user is provided with a symbol such as dynamic icon 28 that shows two puzzle pieces. When the drag box is over a dock-site, i.e., over an existing property sheet system that can accept the property page being dragged, the puzzle pieces join and appear as a second display state 32, thus signifying to the user that the dragged page can be docked e.g., accepted into the that system. FIGURE 5A illustrates the dynamic icon in the first display state 30 wherein the puzzle pieces are separated, indicating that the drag box or geometric shape 20 is not properly positioned for a property page to be docked.

The invention uses the logic depicted in the operational flow chart illustrated in Figure 6. Starting at decision box 34, pressing a mouse key when the cursor is within a property page "tears" the property page and leads along the "yes" branch of box 34 to box 36. As the mouse is dragged, that is, moved with the mouse button pressed, an outline of the property page is drawn on the screen. This process will continue until the user releases the mouse button. Releasing the mouse button leads from along the "yes" branch of box 36 to decision box 38, which determines whether the torn property page can be docked, that is, whether the drag box 20 is positioned over a property page of an existing property sheet system. If docking is possible, the torn page is docked into the "target" system, as in box 42; if the page cannot be docked, a new property sheet system is created for inclusion of the torn property page, as indicated by box 40. Finally, in box 44, the torn page is removed from the original property sheet system, and if no other pages remain in the original, or source, property sheet system, the system is deleted, or destroyed. In the remainder of the specification, the object classes and coding to perform the above-described operations are described.

The present invention uses techniques of object-oriented programming. In object-oriented programming, routines are separate entities that include both data and functions, or methods, for processing the data. Object-oriented programming differs from traditional structured programming, which relied heavily on subroutines, and in which data and the routines that handled the data were usually strictly segregated. Structural programming is organized in terms of computer details, such as files, record, and sorting algorithms; object oriented programming is organized in terms of the physical entities, the "objects," that exist in the environment in which the problem is to be solved. The data and methods stored in a object represent, respectively, the attributes of the physical object and the actions a object is capable of car-

rying out. Objects are related to classes; an object is said to be an instance of a class. A class is not an object, but a template that instructs a compiler how to construct an actual object.

Those of ordinary skill in the art of modern programming are familiar with object-oriented concepts such as encapsulation, inheritance and polymorphism. Encapsulation refers to combining data and functions, or methods, that process the data into a single object, wherein the internal details of the methods and data may be largely hidden. Inheritance refers to a derived class inheriting the features and capabilities of the base, or parent, class. The derived class can then be modified to suit a particular need. Polymorphism allows objects whose properties differ to be able to function together. That is, the interfaces through which objects exchange messages are standardized and not dependent on the exact internal working of an object. Because of encapsulation, polymorphism and inheritance, object oriented-programming is a modular approach that reduces the amount of coding a designer of a new system or function needs to perform. Objects can be joined in a coherent procedure in which objects can be added, modified or deleted without a designer having to rewrite a whole application. A considerable amount of code written by others is reusable, and is often reused. For example, the Microsoft Foundation Class (MFC) Library is a set of more than 100 classes designed to help C++ programmers write Windows-based programs. The present invention built upon classes from the MFC version 4.1 library.

Accordingly, described herein are only those class definitions, and code segments, necessary for one of ordinary skill in the relevant programming arts, possessed of all the teachings herein, to make and use the invention. Classes, objects and codes well known to, or easily created by, those of ordinary skill in the art, or available in published libraries, such as MFC library, are not described, or if described, are only briefly discussed.

The present invention provides for a sophisticated property sheet system that supports modeless operation and tear-away property pages. The system uses the standard system software components illustrated in FIGURE 7. The components include the application frame window 46 and the property sheet system 48. In general, property sheet systems are implemented using three components: a property frame 48A, a property sheet 48B, and one or many property pages 48C. These three components 48A-48C are related. The property pages 48C are the children of the property sheet 48B; the property sheet 48B is the child of the property frame 48A; and the property system 48 is owned by the application's main frame window 46. Because the property frame 48A is typically a top-level window, it generally cannot be a child of another window, but it can be owned.

The present invention extends the standard property sheet system, such as that in FIGURE 7, by extending the standard classes to allow the tearing of property pages, the creation of new property sheet systems, and the docking of torn property pages into existing property sheet systems. The classes are illustrated in FIGURE 8.

The application frame window class 50, labeled CACRFrameWnd in FIGURE 8, provides management support for the property sheet system 55. The application frame window class 50 provides property page dock support, property page availability handling, and property sheet system creation. The application frame window 50 constructs the property sheet system 55 with a creation mask that indicates which property pages 56 to create. Additionally, the application frame window 50 provides the notification interface used by the property sheet system 55 when property or selection information has changed in a property page 36.

The property sheet system 55 is comprised of three distinct parts. The first and least significant of these components is the property sheet frame class 52. The property sheet frame 52 is a mini-frame window that is used to house the property sheet system 55 window. The property sheet frame 52 allows the property sheet system 55 window and pages 56 to be operated in an modeless fashion. The property sheet frame 52 creates the property sheet 54, hides itself when closed, and provides access methods to its owned property sheet 54.

The property sheet class 54 provides management support for the key property pages 56. This support includes property page creation, property page docking support, and property page update facilities.

The individual property pages 56 constitute the final component of the property sheet system 55. Each property page 56 provides specific edit functions for a portion of the data exchanged with an application program. Each property page is derived from a base property page that provides application-specific property page functions. The base property page provides tear-away page support, and facilities to update a keyboard or be updated from a keyboard. The base property page also provides an iterative mechanism for setting property information for multiple keys of keypad annunciator 16 in FIGURE 2 to facilitate disabling of controls.

Further elaboration is provided below, in a separate section on each of the components. Class definitions and a description of the instance variables and methods for the property frame class, property sheet class, and property page classes are provided. Important code sections for the property sheet and property page classes are listed at the end of the specification, and additional code, including code pertaining to modification of the application frame window class is provided in an appendix. Coding for the property sheet frame class is not provided.

55

Application Frame Window Class

Coding for portions of the application frame window class is provided in the appendix.

Property Sheet Frame Class - CPropertySheetFrame:CMiniFrameWnd

The property sheet frame 52 allows the property sheet system window and pages to be operated in a modeless fashion. The property sheet frame 52 can operate to create the property sheet, to hide itself when closed, and provides access methods to its owned property sheet.

As shown in FIGURE 8 and in FIGURE 9A, the CPropertySheetFrame 52 provides the harness window for the property sheet system. The CPropertySheetFrame 52 creates, destroys manages visibility, and facilities modeless operation of the property sheet system.

The CPropertySheetFrame class 52 is defined below in Tables 1 and 2.

10

Table 1 - CPropertySheetFrame Class Definition

Name	Parameters	Returns	Access	Virtua l
CPropertySheetFrame:: ~CPropertySheetFrame	UINT wCreatePageMask	Nothing	public	no
CPropertySheetFrame:: GetPropertySheet	void	Nothing	public	yes
CPropertySheetFrame:: OnCreate	void	CACRPropertyShe et*	public	no
CPropertySheetFrame:: OnClose	LPCREATESTRUCT lpCreateStruct	int	protect ed	no
CPropertySheetFrame:: OnSetFocus	void	void	protect ed	no
CPropertySheetFrame:: OnActivate	CWnd* pOldWnd	void	protect ed	no
	UINT nState, CWnd* pWndOther, BOOL bMinimized	void	protect ed	no

40

Table 2 - CPropertySheetFrame Data Definition

Name	Type	Access
m_pACRPropertySheet	CACRPropertySheet*	private
m_wCreatePageMask	UINT	private

50

Below is a description of the data and methods of the CPropertySheetFrame 52.

Data:

m_pACRPropertySheet: This data member is a pointer to the property sheet system that is created and managed by this frame.

m_wCreatePageMask: This a data member s the initial page creation mask used to define this property sheet system. The page creation mask is a bit field that indicates to the property sheet system which property pages to create.

Methods:

CPropertySheetFrame(UINT wCreatePageMask): This method is the constructor for the CPropertySheetFrame class. The constructor will store the specified page creation mask internally and use it later during the creation process.

5 The constructor will initialize the **m_pACRPropertySheet** member variable to NULL.

~CPropertySheetFrame(): This method is the destructor for the CPropertySheetFrame class. The destructor will destroy the **m_pACRPropertySheet** member variable.

GetPropertySheet(): This method returns the **m_pACRPropertySheet** pointer.

OnCreate(LPCREATESTRUCT lpCreateStruct): This method handles the creation of the frame window. This class 10 customizes the creation process by creating the property sheet system at this time. The property sheet system is created with the page create mask, **m_wCreatePageMask**, that was specified during class creation.

OnClose(): This is a method that intercepts the normal window close behavior. The default behavior of the close process is to destroy the window and this class. This method simply hides the window. The destruction of this class is the responsibility of the CACRFrameWindow.

15 OnSetFocus(CWnd* pOldWnd): This is a method that forwards the focus handling to the embedded property sheet system if one is defined.

OnActivate(UINT nState, CWnd* pWndOther, BOOL bMinimized): This is a method that forwards the activation handling to the embedded property sheet system if one is defined.

20 **Property Sheet Class - CACRPropertySheet:CPropertySheet**

The CACRPropertySheet 54, illustrated in FIGURES 8 and 9B, provides support for the key property pages. CACRPropertySheet customizes the default property sheet behavior by providing automatic property page creation, property page docking support, and property page update facilities. The CACRPropertySheet class 54 is defined below 25 in Tables 3 and 4.

Table 3 - CACRPropertySheet Class Definition

Name	Parameters	Returns	Access	Virtua
CACRPropertySheet	CWnd* pWndParent, UINT wPageCreateFlag	Nothing	public	no
OnCreate	void	Nothing	public	yes
	LPCREATESTRUCT lpCreateStruct	int	protecte d	no
CreatePage	UINT wPageMask	void	public	no
PostNedDestroy	void	void	public	no
DestroyPage	UINT wPageMask	void	public	no
IsPageEnabled	UINT wPageMask	BOOL	public	no
GetPropertyMask	void	UINT	public	no
FindPage	UINT wPageMask	CPropertyPage*	public	no
TearOffPage	UINT wPageMask, LPRECT pTearOffRect	void	public	no
UpdatePageControl	void	void	public	no

Table 4 - CACRPropertySheet Type Definitions

5	Mask	Value
	NO_PAGES	0x00000000
	PAGE_ACTIONS	0x00000001
	PAGES DISPLAYS	0x00000002
10	PAGE_PROGRAMS	0x00000004
	PAGE_COMMANDS	0x00000008
	PAGE_ALARM_LIST	0x00000010
	PAGE_ALARM_COLOR	0x00000020
15	LEGAL_PAGES_ONLY	0x0000003f
	ALL_AVAILABLE_PAGES	0xffffffff
20		

25

Masks:

The following masks are used to define property pages that are used in the CACRPropertySheet implementation.

30

NO_PAGES: Specifies no pages.

PAGE_ACTIONS: Specifies the Alarm Actions Property Page.

PAGES_DISPLAYS: Specifies the Display Property Page.

PAGE_PROGRAMS: Specifies the Program Property Page.

35

PAGE_COMMANDS: Specifies the Command Property Page.

PAGE_ALARM_LIST: Specifies the Alarm List Property Page.

PAGE_ALARM_COLOR: Specifies the Color Property Page.

LEGAL_PAGES_ONLY: Specifies all the legal Property Pages.

ALL_AVAILABLE_PAGES: Specifies all pages, legal or otherwise.

40

Methods:

CACRPropertySheet(CWnd* pWndParent, UINT wPageCreateFlag): This is the constructor for the property sheet system. The constructor will invoke the CreatePage member with the specified page create mask.

45

~CACRPropertySheet() This is the destructor for CACRPropertySheet. The destructor will delete all created property pages.

OnCreate(LPCREATESTRUCT lpCreateStruct): The OnCreate handler automatically snaps this window to the client region of its parent, typically the CPropertySheetFrame.

PostNcDestroy(): This method calls the default PostNcDestroy handler and then deletes itself.

50

CreatePage(UINT wPageMask): This method will creates the property pages specified by the page mask. All created property pages are automatically added to this property sheet.

DestroyPage(UINT wPageMask): This method will find the page with the specified mask, remove it from the property sheet system, and destroy it. The page mask must specify a single page for this method.

55

IsPageEnabled(UINT wPageMask): This method will determine if the page specified by the page mask is present in this property sheet system. This method will return TRUE if the page is found, FALSE otherwise.

GetPageMask(void): This method will return the page mask for all the pages ed in this property sheet system.

FindPage(UINT wPageMask): This method will find the page specified by the page mask. This method will return the specified page if it is found. NULL otherwise.

TearOffPage(UINT wPageMask): This method will request the main frame window to create a property sheet sys-

tem in the specified page mask, remove the specified page from the property sheet system, and if the property page removed is the last on kill this class.

UpdatePageControl(void): This method provides a work around to a MFC 3.0 bug concerning the automatic redraw of the page control used in the property sheet system.

5

Property Page - CACRPropertyPage: CPropertyPage

The CACRPropertyPage 56, as illustrated in FIGURES 8 and 9C, provides the core behavior for the key property sheet system for the process control application program. The key property sheet system is used to configure the keys, 10 of the keypad annunciator 16 in FIGURE 2. CACRPropertyPage 56 provides tear-away page support and facilitates setting of properties from and to a target data source.

The CACRPropertyPage class is defined below in Tables 5 and 6.

15

Table 5 - CACRPropertyPage Class Definition

Name	Parameters	Returns	Access	Virtu al
CACRPropertyPage	UINT wPageID, UINT nIDTemplate, UINT nIDCaption	Nothing	public	no
CACRPropertyPage	LPCSTR lpszTemplateName, UINT nIDCaption	Nothing	public	no
~CACRPropertyPage	void	Nothing	public	yes
GetPageID	void	UINT	public	no
BeginPropertySet	void	void	public	yes
EndPropertySet	void	void	public	yes
GetPropertySheet	void	CACRPropertySheet	public	no
ClipRectToWindow	HWND hWnd, LPRECT pRect	void	protecte d	no
Track	HDC hDC, LPRECT pTrackRect, BOOL bKillTrack	void	protecte d	no
TearOffDialog	LPRECT pTearOffRect	void	protecte d	no
OnLButtonDown	UINT nFlags, CPoint point	void	protecte d	no
OnLButtonUp	UINT nFlags, CPoint point	void	protecte d	no
OnMouseMove	UINT nFlags, CPoint point	void	protecte d	no

Table 6 - CACRPropertyPage Data Definition

Name	Type	Access
m_rTrackRect	RECT	protected
m_bIsTracking	BOOL	protected
m_bIsTearable	BOOL	protected
m_bDitherLast	BOOL	protected
m_sizeLast	SIZE	protected
m_rectLast	RECT	protected
m_ptStartPoint	POINT	protected
m_wPageID	UINT	protected

20

Data:

m_rTrackRect: This variable s the property page tracking rectangle during a page tear operation.

m_bIsTracking: This variable indicates that the property page is in the process of being torn off.

m_bIsTearable: This variable indicates that the current property page is tearable.

m_bDitherLast: This variable indicates the state of the last drag track render operation. To maximize visibility the track rectangle is rendered in a combination of dithered states and non-dithered states.

m_sizeLast: This variable s the size of the last track render operation.

m_rectLast: This variable s the rectangle that was previously rendered during the drag operation.

m_ptStartPoint: This variable s the starting point of the drag operation. The start point will maintain the relative position of the drag rectangle to the mouse.

m_wPageID: This variable s the page id of this property page. The page id is defined on creation of the property page.

35

Methods:

CACRPropertyPage(UINT wPageID,UINT nIDTemplate,UINT nIDCaption): This is the constructor for the CACRPropertyPage dialog. The constructor requires specification of the page ID, a dialog template, and a caption.

CACRPropertyPage(UINT wPageID ,LPCSTR lpszTemplateName,UINT nIDCaption): This is a constructor for the CACRPropertyPage dialog. This version of the constructor allows for the specification of the template using a string.

~CACRPropertyPage(): This is the destructor for the CACRPropertyPage class.

GetPageID(void): This method returns the m_wPageID member variable.

BeginPropertySet(void): This method provides a hook for derived property page classes to initialize data state information to properly reflect data values under multiple keys.

EndPropertySet(void): This method provides a hook for derived property pages to end the property setting process by reflecting collected state information into the property page controls.

GetPropertySheet(void): This method returns the parent property sheet. This method will ASSERT if the parent property sheet is not of CACRPropertySheet style.

ClipRectToWindow(HWND hWnd,LPRECT pRect): This method will clip the specified rectangle to the specified window. The method will maintain the size of the specified rectangle if possible.

Track(HDC hDC,LPRECT pTrackRect,BOOL bKillTrack=FALSE): This method performs tracking for the property page drag/tear operation.

TearOffDialog(LPRECT pTearOffRect): This method performs the property page tear operation. It determines if the specified rectangle represents a tearable state and if so performs the tear operation.

OnLButtonDown(UINT nFlags, CPoint point): This method handles the left mouse button down processing. This method will start the property page tear operation.

OnLButtonUp(UINT nFlags, CPoint point): This method handles the left mouse button up processing. This method will end the property page tear operation.

OnMouseMove(UINT nFlags, CPoint point): This method handles mouse move processing. This method will drag

the page tear rectangle if the mouse button is pressed and the page tear operation is underway.

Key Property Pages

5 In the application described here, wherein the property pages are used to exchange information about, and to configure, the keys of keypad annunciator 16 in FIGURE 2, the key property pages are derived from the CACRPropertyPage base class, as shown in FIGURE 10. The property pages provide specific edit operations for the key data.

10 Each property page will be implemented in a similar fashion. In general, each property page will provide the following interface, as set forth in Table 7.

Table 7 - CACRGenericEditPage Class Definition

Name	Parameters	Returns	Access	Virtual
CACRGenericEditPage	UINT wPageID, UINT nIDTemplate, UINT nIDCaption	Nothing	public	no
CACRGenericEditPage	LPCSTR lpszTemplateName, UINT nIDCaption	Nothing	public	no
~CACRGenericEditPage	void	Nothing	public	yes
BeginPropertySet	void	void	public	yes
EndPropertySet	void	void	public	yes

35 Code sections for the property sheet and property page classes, as well as code sections pertaining to modification of the application frame window class, are provided in an appendix. The code is written in the C++ programming language, but can be used as the operational foundation for other implementations.

40

45

50

55

APPENDIX

5

10

15

C++ Code for CACRPropertySheet Class

20

```

//*****.
//*****
// Class:      CreatePage
// Description: Create the property page for this mask and add it to the system.
//              This routine will create multiple pages if multiple masks are
//              specified.
//
// Returns:     Bool
//*****.

1 CModelessPropertySheet::CreatePage(UINT wPageMask)
30
CPropertyPage* pPage;

//-----
// Determine if the individual page types
// are defined and create and add them to the
35

40
        // sheet.
        //-----
if (wPageMask & PAGE_ACTIONS)
{
    pPage = new CAImActions();
    AddPage(pPage);
}
45
if (wPageMask & PAGE_DISPLAYS)
{
    pPage = new COdisplayListPage();
    AddPage(pPage);
}

```

50

© 1996, The Foxboro Company

55

```

}
if (wPageMask & PAGE_PROGRAMS)
{
    pPage = new CProgramListPage();
    AddPage(pPage);
}
if (wPageMask & PAGE_COMMANDS)
{
    pPage = new CCommandDlg();
    AddPage(pPage);
}
if (wPageMask & PAGE_ALARM_LIST)
{
    pPage = new CALarmList();
    AddPage(pPage);
}
if (wPageMask & PAGE_ALARM_COLOR)
{
    pPage = new CNewAlarmColor();
    AddPage(pPage);
}

15

//*****
//***** Method:      DestroyPage
//***** Description: This method toasts the property page associated with this mask.
//*****          This is used when a property page has been removed from one sheet
//*****          and put into another sheet.
//***** Returns:     Bool
//*****          -----
void CModelessPropertySheet::DestroyPage(UINT wPageMask)
{
    CPropertyPage* pPropertyPage;
    //-----  

    //-----  

    //-----  

    if ((pPropertyPage = FindPage(wPageMask)) != NULL)
    {
        RemovePage(pPropertyPage);
        delete pPropertyPage;
    }
}

30

//*****
//***** Method:      FindPage
//***** Description: Get the property page pointer based on a mask
//***** Returns:     Bool
//*****          -----
CPropertyPage* CModelessPropertySheet::FindPage(UINT wPageMask)
{
    CACRPropPage* pPage = NULL;
    for (int i=0; i<GetPageCount(); i++)
    {
        // Am I the guy we want?

45

//*****
//***** Method:      CModelessPropertySheet
//***** Description: This defines the core modeless property sheet.
//*****          -----
}

50

```

Copyright 1996, The Foxboro Company

```

5      #define NO_PAGES          0x00000000
      #define PAGE_ACTIONS      0x00000001
      #define PAGE_DISPLAYS     0x00000002
      #define PAGE_PROGRAMS     0x00000004
      #define PAGE_COMMANDS     0x00000008
      #define PAGE_ALARM_LIST   0x00000010
      #define PAGE_ALARM_COLOR   0x00000020

10     #ifdef SOFTWARE
      #define LEGAL_PAGES_ONLY  0x0000003f
      #define ALL_AVAILABLE_PAGES 0xffffffff
      #else
      #define LEGAL_PAGES_ONLY  0x0000001f
      #define ALL_AVAILABLE_PAGES 0xffffffffdf
      #endif

15     class CModelessPropertySheet;
      class CKeyboardCore;
      class CACRKey;

      class CModelessPropertySheet : public CPropertySheet
      {
      private:
          CKeyboardCore* m_pOwnerKeyboard;

      public:
          DECLARE_DYNAMIC(CModelessPropertySheet)
          CModelessPropertySheet (CWnd* pWndParent,UINT wPageCreateFlag);
          ~CModelessPropertySheet ();

20        // Attributes
          void           CreatePage           (UINT wPageMask);
          void           DestroyPage          (UINT wPageMask);
          BOOL          IsPageEnabled        (UINT wPageMask);
          UINT          GetPageMask         (void);
          CPropertyPage* FindPage           (UINT wPageMask);
          void           TearOffPage         (UINT wPageMask,LPRECT pTearOffRect);
          void           UpdatePageControl   (void);

25        void           SetPropertiesFromKeyboard (CKeyboardCore* pKeyboard);

          if ( !(pPage = (CACRPropPage*)GetPage(i)) != NULL) &&
              (pPage->IsKindOf(RUNTIME_CLASS(CACRPropPage))) &&
              (pPage->GetPageID() & wPageMask) )
          {
              return pPage;
          }

30        ASSERT(pPage != NULL);           // If we are asking for a NULL page some
                                         // bogus is going on.
          return pPage;
      }

35        //*****
// Class:      IsPageEnabled
// Description: This routine loops through the pages associated with this
//               dialog system and determines if they are present.
// Returns:    Bool
//*****
40        BOOL CModelessPropertySheet::IsPageEnabled(UINT wPageMask)
      {
          CACRPropPage* pPage = NULL;

          for (int i=0; i<GetPageCount(); i++)
          {
              // Am I the guy we want?
              if ( !(pPage = (CACRPropPage*)GetPage(i)) != NULL) &&
                  (pPage->IsKindOf(RUNTIME_CLASS(CACRPropPage))) &&
                  (pPage->GetPageID() & wPageMask) )
              {
                  return TRUE;
              }
          }
          return FALSE;
      }

```

50

Copyright 1996, The Foxboro Company

55

```

5 //*****
// Class:      GetPageMask
// Description: This routine loops through the pages associated with this
// dialog system and collects their page masks.
// Returns:     UINT
//*****  

10 UINT CModelessPropertySheet::GetPageMask(void)
{
    UINT          wPageMask = 0;
    CACRPropPage* pPage = NULL;
    for (int i=0; i<GetPageCount(); i++)
    {
        // Am I the guy we want?
        if ( ((pPage = (CACRPropPage*)GetPage(i)) != NULL) &&
            (pPage->IsKindOf(RUNTIME_CLASS(CACRPropPage)) ) )
            wPageMask |= pPage->GetPageID();
    }
    return wPageMask;
15  

//*****
// Class:      PageChanged
// Description: This routine is called by the Property pages when something
// has occurred that changes their data. We will tell the keyboard
// core so he may update his information.
// Returns:     Nothing
20  

25 //*****
// Method:     TearOffPage
// Description: This routine removes the requested page from this sheet
// and tells the frame to create a new one.
// Returns:     Nothing
30 //*****
d CModelessPropertySheet::TearOffPage(UINT wPageMask, LPRECT pTearOffRect)
CFrameWnd* pNewFrame;
if ((pNewFrame = (CFrameWnd*)ACRGetMainWindow()->AddPropertySheet(GetOwnerKeyboard(), wPageMask, pTearOffRect)) != NULL)
{
35  

    //-----
    // If it is the last page we should commit
    // suicide
    //-----
    if (GetPageCount() == 1)
        GetParent()->DestroyWindow();
40    //-----
    // Get It. Remove It. KILL It.
    //-----
    else
    {
        DestroyPage(wPageMask);
        //-----
        // Put the new one on top
        //-----
45    if (pNewFrame != (CFrameWnd*)GetParent())
        pNewFrame->BringWindowToFront();
    }
}
50

```

Copyright 1996, The FOXBORO Company

```

5 //*****
// Method:      UpdatePageControl
// Description: This method forces the control used in the property pages to
//               be redrawn.
// Returns:     Nothing
//*****
10 void CModelessPropertySheet::UpdatePageControl(void)
{
    m_tabRow.SetCurSel(m_tabRow.GetItemCount()-1);
}

```

C++ Code for CACRPropertyPage Class

```

15
20 //*****
// Class:      CACRPropPage
// Description: This defines the alarm color dialog.
//*****
25 class CKeyboardCore;
class CACRKey;
class CModelessPropertySheet;
class CACRDocument;

class CACRPropPage : public CPropertyPage
25     DECLARE_DYNAMIC(CACRPropPage)

protected:
    RECT          m_rTrackRect;
    BOOL          m_bIsTracking;
    BOOL          m_bIsTeakable;
    BOOL          m_bDitherLast;
    SIZE          m_sizeLast;
    RECT          m_rectLast;
    POINT         m_ptStartPoint;
    UINT          m_wPageID;           // System defined user mask
    BOOL          m_bEnableTracking;
    BOOL          m_SelectedKeyCount;
    BOOL          m_bIsInitiated;

    // Construction
public:
    CACRPropPage (UINT wPageID,UINT nIDTemplate,UINT nIDCaption=0);
    CACRPropPage (LPCSTR lpszTemplateName,UINT nIDCaption);
    ~CACRPropPage();

    virtual BOOL  OnSetActive ();
    virtual BOOL  CanClose ();
    virtual INT   GetPageID ();
    virtual void  SetPropertiesFromKeyboard (CKeyboardCore* pKeyboard);
    virtual void  SetKeyboardFromProperties (CKeyboardCore* pKeyboard, LONG lHint=0L);

    virtual void  BeginPropertySet ();
    virtual void  SetPropertyFromKey (CACRKey* pKey);
    virtual void  SetKeyFromProperty (CACRKey* pKey, LONG lHint=0L);
    virtual void  EndPropertySet ();
    virtual INT   GetUndoOperationID (LONG lHint) {return STR_ID_EDIT_UNDO;};

45    CModelessPropertySheet* GetPropertySheet () {void;};
    CACRDocument*  GetDocument () {void;};
    CKeyboardCore* GetKeyboard () {void;};

    // Dialog Data
    //{{AFX_DATA(CACRPropPage)

```

50

Copyright 1996, The Foxboro Company

55

```

    //)AFX_DATA

    // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CACRPropPage)
protected:
    //}}AFX_VIRTUAL

    // Implementation
protected:
    void            ClipRectToWindow      (HWND hWnd,LPRECT pRect);
    void            Track                (HDC hDC,LPRECT pTrackRect,BOOL bKillTrack=FALSE),
    void            TearOffDialog        (LPRECT ptearOffRect);

    // Generated message map functions
    //{{AFX_MSG(CACRPropPage)
    virtual BOOL    OnInitDialog        ();
    afx_msg void    OnLButtonDown       (UINT nFlags, CPoint point);
    afx_msg void    OnLButtonUp         (UINT nFlags, CPoint point);
    afx_msg void    OnMouseMove         (UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

inline UINT CACRPropPage::GetPageID(void)
{
    return m_wPageID;
}
#endif

//*****
// Method:      CACRPropPage::CACRPropPage
// Description: This routine constructs the ACFG Property Page
// Returns:     Nothing
//*****


CACRPropPage::CACRPropPage(UINT wPageID,UINT nIDTemplate,UINT nIDCaption) : CPropertyPage(nIDTemplate,nIDC
on)
{
    m_wPageID          = wPageID;
    m_bEnableTracking = FALSE;
    m_bIsTracking     = FALSE;
    m_bIsTearable     = FALSE;
    m_bDitherLast     = FALSE;
    m_sizeLast.cy      = 0;
    ::SetRectEmpty(&m_rectLast);
    m_SelectedKeyCount = 0;
    m_bIsInitiated    = FALSE;

    //{{AFX_DATA_INIT(CACRPropPage)
    // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}

//*****
// Method:      CACRPropPage::CACRPropPage
// Description: This routine constructs the ACFG Property Page
// Returns:     Nothing
//*****


CRPropPage::CACRPropPage(LPCSTR lpszTemplateName,UINT nIDCaption) : CPropertyPage(lpszTemplateName,nIDC
{
    //{{AFX_DATA_INIT(CACRPropPage)
    // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}

50

```

Copyright 1996, The Foxboro Company

```

5
//*****
// Class:      OnLButtonDown
// Description: Tear away this popup and make a modeless dialog out of it .
//
// Returns:     BOOL
//*****
void CACRPropPage::OnLButtonDown(UINT nFlags, CPoint point)
{
    HWND    hDesktop = ::GetDesktopWindow();
10
    if (GetParent()->IsWindowVisible() && hDesktop)
    {
        HDC    hDC = ::GetDCEx(hDesktop,NULL,DCX_PARENTCLIP);
        // Render the popup outline
        GetClientRect(&m_rTrackRect);
        ::MapWindowPoints(GetSafeHwnd(),hDesktop,(LPPOINT)&m_rTrackRect,2);
        m_ptStartPoint = point;
        m_bIsTracking = TRUE;
        m_bIsTearable = FALSE;
        m_bDitherLast = FALSE;
        ::SetRectEmpty(&m_rectLast);

        m_sizeLast.cx = m_sizeLast.cy = 0;
        ::SetCapture(GetSafeHwnd());
20
        // Render the popup outline
        Track(hDC,&m_rTrackRect);

        // Release the display context
        ::ReleaseDC(hDesktop,hDC);
    }
    else
25
        CPropertyPage::OnLButtonDown(nFlags, point);
    }

    //*****
    // Class:      OnLButtonUp
    // Description: If tracking a popup make a modeless dialog out of it.
    //
    // Returns:     BOOL
    //*****
30
void CACRPropPage::OnLButtonUp(UINT nFlags, CPoint point)
{
35
    if (m_bIsTracking)
    {
        RECT    rClientRect;
        GetClientRect(&rClientRect);
        HWND    hDesktop = ::GetDesktopWindow();
        HDC    hDC;

        ASSERT(hDesktop);
40
        hDC = ::GetDCEx(hDesktop,NULL,DCX_PARENTCLIP);
        // Undraw and release dc
        Track(hDC,&m_rTrackRect,TRUE);

        ::ReleaseDC(hDesktop,hDC);

45
        // Create the modeless property
        // sheet dialog here.
        if ((m_bIsTearable) && !::PtInRect(&rClientRect,m_ptStartPoint))
            TearOffDialog(&m_rTrackRect);
    }

    //*****
    // Class:      OnRButtonDown
    // Description: If tracking a popup make a modeless dialog out of it.
    //
    // Returns:     BOOL
    //*****
50
}

```

Copyright 1996, The Foxboro Company

```

    m_bIsTracking = FALSE;
    m_bIsTearable = FALSE;
    ::ReleaseCapture();
}
else
    CPropertyPage::OnLButtonUp(nFlags, point);

//*****
//***** Class:      OnMouseMove
//***** Description: Track the popup if it being torn away.
//***** Returns:     BOOL
//***** *****

void CACRPropPage::OnMouseMove(UINT nFlags, CPoint point)
{
    RECT rClientRect;
    GetClientRect(&rClientRect);
    if (m_bIsTracking)
    {
        HWND hDesktop = ::GetDesktopWindow();
        HDC hDC;

        ASSERT(hDesktop);
        hDC = ::GetDCEx(hDesktop, NULL, DCX_PARENTCLIP);

        if (!::PtInRect(&rClientRect, point))
        {
            int nDX, nDY;
            if (m_bIsTearable)
                Track(hDC, &m_rTrackRect); // Undraw
        }
        else
            // Recalculate the tracking rectangle
            nDX = point.x - m_ptStartPoint.x;
            nDY = point.y - m_ptStartPoint.y;
            ::OffsetRect(&m_rTrackRect, nDX, nDY);
            ClipRectToWindow(hDesktop, &m_rTrackRect);
            m_ptStartPoint = point; // Remeber this point
    }
    Track(hDC, &m_rTrackRect); // Redraw
    ::ReleaseDC(hDesktop, hDC); // Release
    m_bIsTearable = TRUE;
}
else if (m_bIsTearable)
{
    Track(hDC, &m_rTrackRect, TRUE); // Undraw
    m_bIsTearable = FALSE;
}
else
    CPropertyPage::OnMouseMove(nFlags, point);
}

#define CX_BORDER 1
#define CY_BORDER 1
#define TITLEHEIGHT 16
//*****
//***** Class:      Track
//***** Description: Render the tracking rectangle in XNOR mode on the specified
//***** display context.
//***** Returns:     BOOL
//***** *****

50

```

Copyright 1996, The Foxboro Company

```

void CACRPropPage::Track(HDC hDC, LPRECT pTrackRect, BOOL bKillTrack)
{
    CDC* pDC=CDC::FromHandle(hDC);
    // default to thin frame
    CSize size(CX_BORDER, CY_BORDER);
    // determine new rect and size
5     CRect rect;
    CBrush* pWhiteBrush = CBrush::FromHandle((HBRUSH)::GetStockObject(WHITE_BRUSH));
    CBrush* pDitherBrush = CDC::GetHalftoneBrush();
    CBrush* pBrush = pWhiteBrush;
    HCURSOR hCursor;

    // Set track rect size
    rect = *pTrackRect;

10    // Use thick frame if we are not over a docking
        // site
    if (ACRGetMainWindow()->CanDock(&rect) == NULL)
    {
        size.cx = GetSystemMetrics(SM_CXFRAME) - CX_BORDER;
        size.cy = GetSystemMetrics(SM_CYFRAME) - CY_BORDER;

15    pBrush = pDitherBrush;
        hCursor = AfxGetApp()->LoadCursor(IDC_NOTDOCKED);
    }
    else
        hCursor = AfxGetApp()->LoadCursor(IDC_DOCKED);

20    if (bKillTrack || !m_bIsTearable)
    {
        size.cx = size.cy = 0;
        hCursor = AfxGetApp()->LoadStandardCursor(IDC_ARROW);
    }
    ::SetCursor(hCursor);

25    // draw it and remember last size
    pDC->DrawDragRect(&rect, size, &m_rectLast, m_sizeLast,
                       pBrush, m_bDitherLast ? pDitherBrush : pWhiteBrush);
    m_rectLast = rect;
    m_sizeLast = size;
    m_bDitherLast = (pBrush == pDitherBrush);
}

//*****
//***** Class:      ClipRectToWindow
//***** Description: This routine clips the specified rectangle to the specified
//***** window.
//***** Returns:     BOOL
//***** *****

30 void CACRPropPage::ClipRectToWindow(HWND hWnd, LPRECT pRect)
{
    RECT rClientRect;
    int nWidth,nHeight;

35    #ifndef WU_APP
        ::GetClientRect(hWnd,&rClientRect);
        .se
        ::GetWindowRect(hWnd,&rClientRect);
    #endif
    nWidth = pRect->right - pRect->left;
    nHeight = pRect->bottom - pRect->top;
40    // Clip top and bottom

    if (pRect->top < rClientRect.top)
    {
        pRect->top = rClientRect.top;
        pRect->bottom = pRect->top + nHeight;
    }
    else if (pRect->bottom > rClientRect.bottom)
45    {
        pRect->bottom = rClientRect.bottom;
        pRect->top = pRect->bottom - nHeight;
    }
}

```

50

Copyright 1996, The Foxboro Company

55

```

      // Clip left and right
    if (pRect->left < rClientRect.left)
    {
        pRect->left = rClientRect.left;
        pRect->right = pRect->left + nWidth;
    }
    else if (pRect->right > rClientRect.right)
    {
        pRect->right = rClientRect.right;
        pRect->left = pRect->right - nWidth;
    }
}

10
//*****
//*****
// Class:      TearOffDialog
// Description: This routine creates another version of this dialog, only in
//               modeless form.
// Returns:     BOOL
//*****
#define MIN_NEW_DELTAX 5
#define MIN_NEW_DELTAY 5

20
    d CACRPropPage::TearOffDialog(LPRECT pTearOffRect)
{
    HWND             hDesktop = ::GetDesktopWindow();
    RECT            rClientRect;
    CModelessPropertySheet* pPropertySheet = GetPropertySheet();

    //-----
    // Convert the client coords to the
    // desktop coords to determine if the
    // tracking window has moved enough to
    // tear it off.
    //-----

25
    GetClientRect(&rClientRect);
    ::MapWindowPoints(GetSafeHwnd(), hDesktop, (LPOINT)&rClientRect, 2);

    //-----
    // Get this pages ID, create a new property
    // sheet with it, and destroy the page.
    //-----

30
    if (pPropertySheet)
        pPropertySheet->TearOffPage(GetPageID(), pTearOffRect);
}

35
//*****
//*****
// Class:      GetPropertySheet
// Description: This routine returns the parent property sheet in a digestable
//               format.
// Returns:     BOOL
// *****

40
CModelessPropertySheet* CACRPropPage::GetPropertySheet(void)
{
    CWnd*           pHnd = GetParent();
    CModelessPropertySheet* pPropertySheet;

45
    pPropertySheet = (CModelessPropertySheet*)pHnd;
    if (pPropertySheet && (pPropertySheet->IsKindOf(RUNTIME_CLASS(CModelessPropertySheet)))) )
        return pPropertySheet;

    return NULL;
}

```

50

Copyright 1996, The Foxboro Company

55

C++ Code for Application Frame

```

5   //*****
6   // Method:      AddPropertySheet
7   // Description: This routine creates a frame window with property sheet, and
8   //                populates it with available property pages.
9   //
10  // Returns:     CPropertySheetFrame*
11  //
12  //*****
```

```

CPropertySheetFrame* CACRMainFrame::AddPropertySheet(CKeyboardCore* pActiveKeyboard,UINT wPageMask,LPRECT pPro
13  pertyRect)
14  {
15      CPropertySheetFrame* pPropertySheetFrame=NULL;
16
17      // A valid keyboard is required
18      ASSERT(pActiveKeyboard != NULL);
19
20      // If the user is creating a standard property frame
21      // we should give them all the available pages
22      if (wPageMask == ALL_AVAILABLE_PAGES)
23          wPageMask = GetAvailablePageMask();
24
25      if (wPageMask == NO_PAGES)
26          return FALSE;
27
28      // Determine if we can dock this sheet to an existing
29      // frame.  We are in docking mode if the pPropertyRect is
30      // specified.
31      if (lpPropertyRect != ((pPropertySheetFrame=CanDock(pPropertyRect)) != NULL))
32      {
33          CModelessPropertySheet* pPropertySheet = pPropertySheetFrame->GetPropertySheet();
34          pPropertySheet->CreatePage(wPageMask);
35
36          #ifndef DONTRINIT
37              pPropertySheet->SetOwnerKeyboard(pActiveKeyboard);
38              pPropertySheet->SetPropertiesFromKeyboard(pActiveKeyboard);
39          #endif
40          pPropertySheet->UpdatePageControl();
41      }
42      else
43      {
44          // Watch out for those exceptions
45          TRY
46          {
47              CRect             rect(0,0,0,0);
48              CString           strTitle;
49
50              // Create this guy and position it.
51              pPropertySheetFrame = new CPropertySheetFrame(wPageMask);
52
53              strTitle.Empty();
54              if (!pPropertySheetFrame->Create(NULL, strTitle, WS_POPUP | WS_CAPTION | WS_SYSMENU, rect, this))
55                  delete pPropertySheetFrame;
56
57              UpdateKeyPropertyTitle(pPropertySheetFrame,pActiveKeyboard);
58
59              // Initialize the data
60              pPropertySheetFrame->GetPropertySheet()->SetOwnerKeyboard(pActiveKeyboard);
61              pPropertySheetFrame->GetPropertySheet()->SetPropertiesFromKeyboard(pActiveKeyboard);
62
63              // If a property rect is specified use it to
64              // position this guy
65              if (lpPropertyRect)
66                  pPropertySheetFrame->SetWindowPos(NULL,lpPropertyRect->left,lpPropertyRect->top,0,0,SWP_NOSIZE|S
67                  WP_NOCOPY|WP_NOZORDER);
68              else
69                  pPropertySheetFrame->CenterWindow(ACRGetDialogCenterWindow());
70
71              if (!pPropertySheetFrame->IsWindowVisible())
72                  pPropertySheetFrame->ShowWindow(SW_SHOW);
73
74          }
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
813
814
815
815
816
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
838
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
858
858
859
859
860
861
862
863
864
865
866
867
867
868
869
869
870
871
872
873
874
875
876
876
877
877
878
879
879
880
881
882
883
884
885
886
886
887
888
888
889
889
890
891
892
893
894
894
895
895
896
896
897
897
898
898
899
899
900
901
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
161
```

```

5
    CATCH_ALL(e)
    {
        delete pPropertySheetFrame;
        pPropertySheetFrame = NULL;
    }
    END_CATCH_ALL
    return pPropertySheetFrame;
}

10
//*****
//*****
// Method:      GetAvailablePageMask
// Description: This routine calculates the available page masks by enumerating
//               all the child property frames.
//
// Returns:     UINT
//*****
15
UINT CACRMainFrame::GetAvailablePageMask(void)
{
    UINT    wPageMask = LEGAL_PAGES_ONLY;
    CWnd*   pWnd = this;

    // Loop through all the child window and find the
    // frames and add their Page Mask to this page mask.
20
    while (pWnd)
    {
        if ( (pWnd->IsKindOf(RUNTIME_CLASS(CPropertySheetFrame))) &&
            (pWnd->GetParent() == GetSafeHwnd()) )
        {
            CPropertySheetFrame*   pPropertyFrame = (CPropertySheetFrame*)pWnd;
            CModelessPropertySheet* pPropSheet = pPropertyFrame->GetPropertySheet();

            wPageMask |= -pPropSheet->GetPageMask();
        }
        pWnd = pWnd->GetNextWindow(GW_HWNDPREV);
    }
    return wPageMask;
}

25

30
//*****
//*****
// Method:      CanDock
// Description: This routine determines if we dock the specified frame to
//               an existing frame.
//
// Returns:     CPropertySheetFrame* -> Docking frame if one is found.
//*****
35
#define DOCKX_FUZZ 10
#define DOCKY_FUZZ 10

CPropertySheetFrame* CACRMainFrame::CanDock(LPRECT pPropertyRect)
{
    CPropertySheetFrame*   pPropertySheetFrame=NULL;
    UINT                  wPageMask = LEGAL_PAGES_ONLY;
    CWnd*                pWnd = this;
    RECT                 rTestRect = *pPropertyRect;

    // reduce the test rect
    ::InflateRect(&rTestRect,-DOCKX_FUZZ,-DOCKY_FUZZ);

    // Loop through all the child window and find the
    // frames and add their Page Mask to this page mask.
40
    while (pWnd)
    {
        if ( (pWnd->IsKindOf(RUNTIME_CLASS(CPropertySheetFrame))) &&
            (pWnd->GetParent() == GetSafeHwnd()) )
        {
            RECT      rWindowRect;

```

Copyright 1996, The Foxboro Company

```

    // Get the window rect for this frame;
    pHnd->GetWindowRect(&rWindowRect);
    if ( (rWindowRect.top < rTestRect.top) && (rWindowRect.bottom > rTestRect.bottom) &&
        (rWindowRect.left < rTestRect.left) && (rWindowRect.right > rTestRect.right) )
5       return (CPropertySheetFrame*)pHnd;
    } pHnd = pHnd->GetNextWindow(GW_HWNDPREV);
    return NULL;
}

10  //*****
//***** SyncPropertySheets
// Method:      SyncPropertySheets
// Description: This routine syncs the property sheets up with the current
//               selection state of the keyboard.
// Returns:     Nothing
15  //*****
void CACRMainFrame::SyncPropertySheets(CKeypadCore* pKeyboard)
{
    // Get the active keyboard if one is not specified.
    if (!pKeyboard)
        pKeyboard = GetActiveKeyboard();

20    // Sync each property frame if a keyboard is
        // specified.
    if (pKeyboard)
    {
        FRAMEPOSITIONINFO FramePositionInfo;

        FramePositionInfo.hWindowOnTopOf = HWND_TOP;
        FramePositionInfo.hDeferPositionInfo = ::BeginDeferWindowPos(10);
        FramePositionInfo.hParent = GetSafeHwnd();
        FramePositionInfo.pKeyboard = pKeyboard;
        FramePositionInfo.bInvalidate = FALSE;
        FramePositionInfo.pScreenRect = NULL;
        FramePositionInfo.wPropMode = kPropSync;

        ::EnumWindows(ACREnumChildProc, (LPARAM)&FramePositionInfo);

        if (FramePositionInfo.hDeferPositionInfo)
            ::EndDeferWindowPos(FramePositionInfo.hDeferPositionInfo);
30    }
}

35  //*****
//***** ClosePropertySheets
// Method:      ClosePropertySheets
// Description: This method enumerates each property sheet and closes them.
// Returns:     Nothing
//*****
void CACRMainFrame::ClosePropertySheets(void)
{
    // Sync each property frame if a keyboard is
    // specified.
    FRAMEPOSITIONINFO FramePositionInfo;

    memset(&FramePositionInfo, 0, sizeof(FramePositionInfo));

    FramePositionInfo.hWindowOnTopOf = HWND_TOP;
    FramePositionInfo.hDeferPositionInfo = NULL;
    FramePositionInfo.hParent = GetSafeHwnd();
    FramePositionInfo.pKeyboard = NULL;
    FramePositionInfo.bInvalidate = FALSE;
    FramePositionInfo.pScreenRect = NULL;
    FramePositionInfo.wPropMode = kPropClose;

    ::EnumWindows(ACREnumChildProc, (LPARAM)&FramePositionInfo);
}

```

50

Copyright 1996, The Foxboro Company

55

Claims**1. Apparatus for computerised data processing, comprising**

5 processor means for executing a program,
 a memory element for storing the program,
 a display monitor for providing a display,
 selection means for selecting a property page of a first property sheet system, the first property sheet system
 including at least one property page displayable at a first position on the display monitor, and
 10 creation means for creating a second property sheet system including said selected property page, said
 selected property page being displayable at a second display position on the display monitor.

**2. Apparatus for configuring a peripheral via a computer or configuring the intercommunication between a computer
 and a peripheral, the computer comprising;**

15 processor means for executing a program,
 a memory element for storing at least the program,
 a display monitor for providing a display,
 the processor means being adapted to display via the display monitor property sheet systems for receiving
 20 configuration settings data from a user, the processor means being further adapted to store said data in the
 memory element at least temporarily and to employ said data when communicating with the peripheral;
 selection means for selecting a property page of a first property sheet system, the first property sheet system
 including at least one property page displayable at a first position on the display monitor, and
 25 creation means for creating a second property sheet system including said selected property page, said
 selected property page being displayable at a second display position on the display monitor.

3. The apparatus of Claim 1 or Claim 2 further including

30 removal means for removing said property page selected by said selection means from said first property
 sheet system such that said selected property page is not displayable by said first property sheet system.

4. The apparatus of Claim 3 further including

35 destruction means for destroying said first property sheet system when said first property sheet system does
 not include any property pages.

5. The apparatus of any preceding Claim further comprising

40 positioning means for varying and indicating on the display monitor said second display position at which said
 property page selected by said selection means and included in said second property sheet system is display-
 able by said second property sheet system.

6. The apparatus of any preceding claim further including

45 docking means for placing said property page selected by said selection means into a selected property sheet
 system for inclusion therein, said property page being displayable on the display monitor by said selected prop-
 erty sheet system.

7. The apparatus of Claim 6 further comprising

50 positioning means for varying and indicating the position of the display monitor of a symbol representative of a
 property page selected by said selection means, and
 proximity means for determining a parameter responsive to the distance on the display monitor between said
 symbol and the position on the display monitor of a displayed property page of a property sheet system, and
 55 wherein said parameter is less than a first selected value, said docking means in response to a first user input
 places said selected property page in said property sheet system displaying said displayed property page for
 inclusion therein and for display on the display monitor thereby.

8. The apparatus of Claim 7 wherein when said parameter is greater than a second selected value, said creation

means in response to a second user input displays said selected property page included in said second property sheet system at a position on the display monitor determined by the display position of said symbol.

9. The apparatus of Claim 7 further comprising

5 icon means for displaying on the display monitor an icon having at least a first display state, said icon displayed in said first display state when said parameter is less than said first selected value, said first display state for indicating to a user that said docking means, in response to said first user input, will place said selected property page into said property sheet system displaying said displayed property page.

10 10. The apparatus of Claim 9 wherein said icon means further comprises means for displaying said icon in a second display state when said parameter is greater than a second selected amount, said second display state for indicating to a user that said second property sheet system will display said property page selected by said selection means at a position on the display monitor determined by the position of said symbol.

15 11. A method for displaying property pages on a computer display monitor comprising the steps of

20 creating a first property sheet system including at least one property page displayable at a first position on a display monitor,

selecting a property page of said first property sheet system, and

25 creating a second property sheet system including said selected property page, said selected property page being displayable at a second display position on the display monitor.

12. A method of displaying on a computer display monitor configuration settings governing a peripheral or intercommunication between a computer and a peripheral, comprising the steps of

25 creating a first property sheet system including at least one property page displayable at a first position on a display monitor,

selecting a property page of said first property sheet system, and

30 creating a second property sheet system including said selected property page, said selected property page being displayable at a second display position on the display monitor.

13. The method of Claim 11 or Claim 12 further including the steps of

35 selecting a property page from one of said first and second property sheet systems, and

docking said property page selected from one of said first and second property sheet systems into the other of said first and second property sheet systems for display thereby.

14. A method for displaying property pages on a display monitor, comprising the steps of

40 creating first and second property sheet systems, said first property sheet system including a property page displaying at first display position on the display monitor,

selecting a property page of said first property sheet system, and

45 docking said selected property page into said second property sheet system, said selected property page displayable at a second display position on the display monitor.

50

55

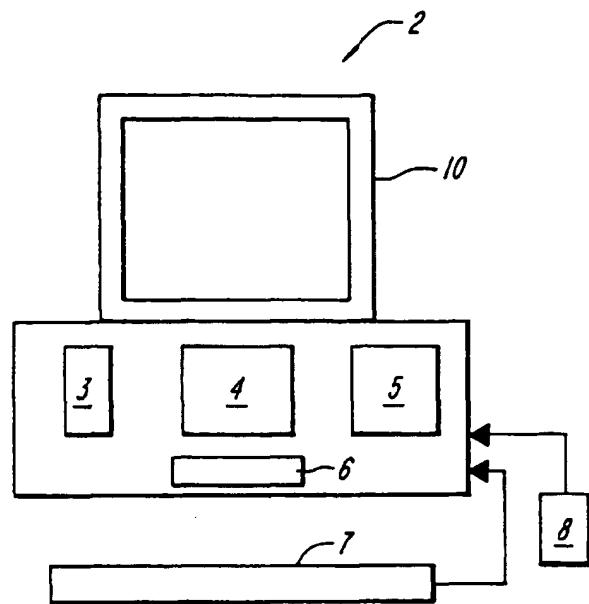


FIG. 1

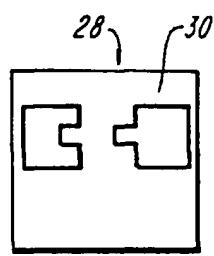


FIG. 5A

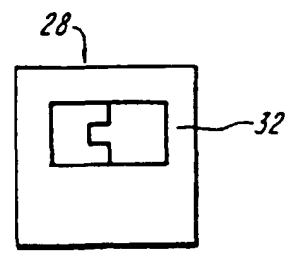


FIG. 5B

FIG. 2

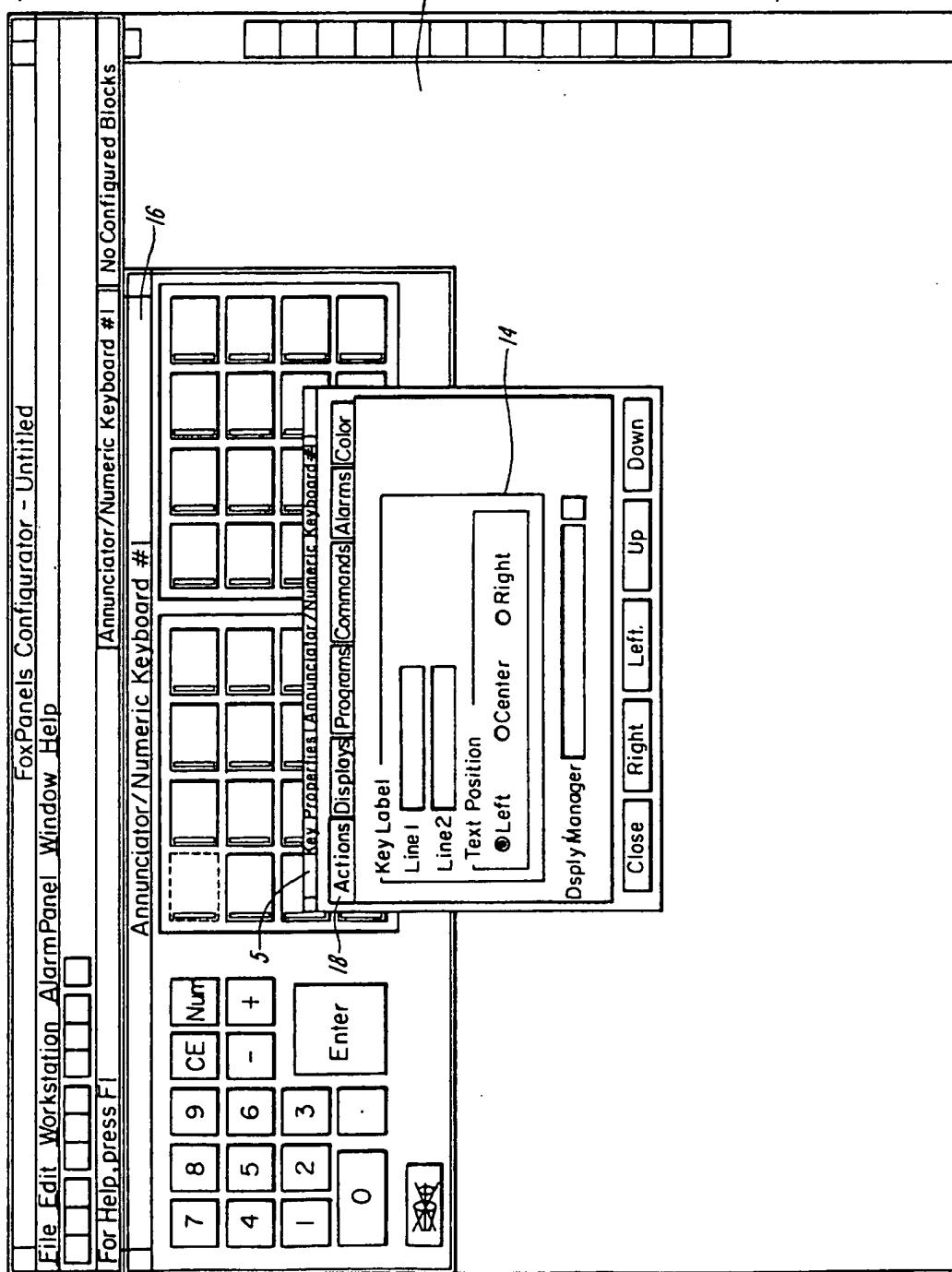


FIG.
3

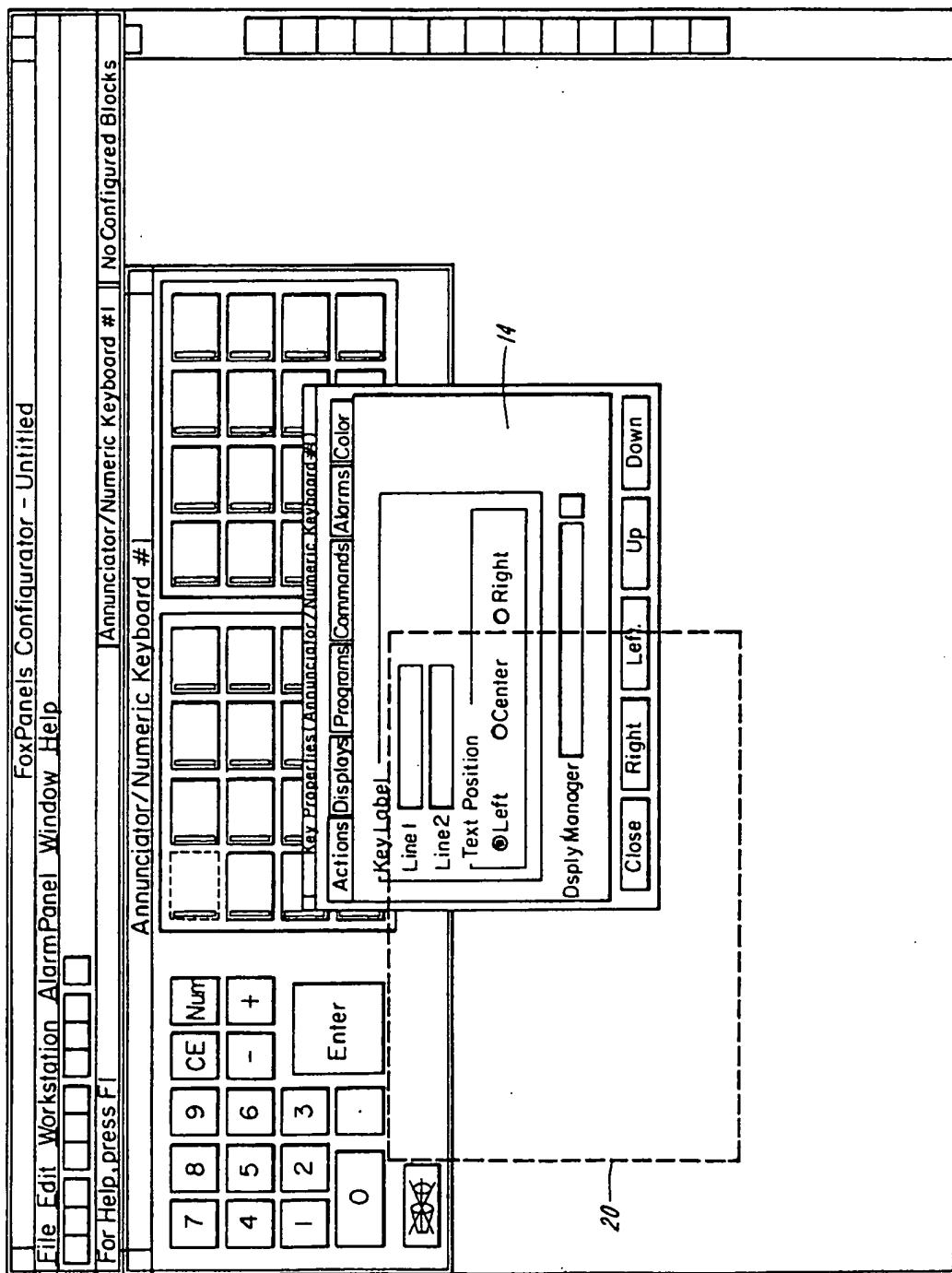
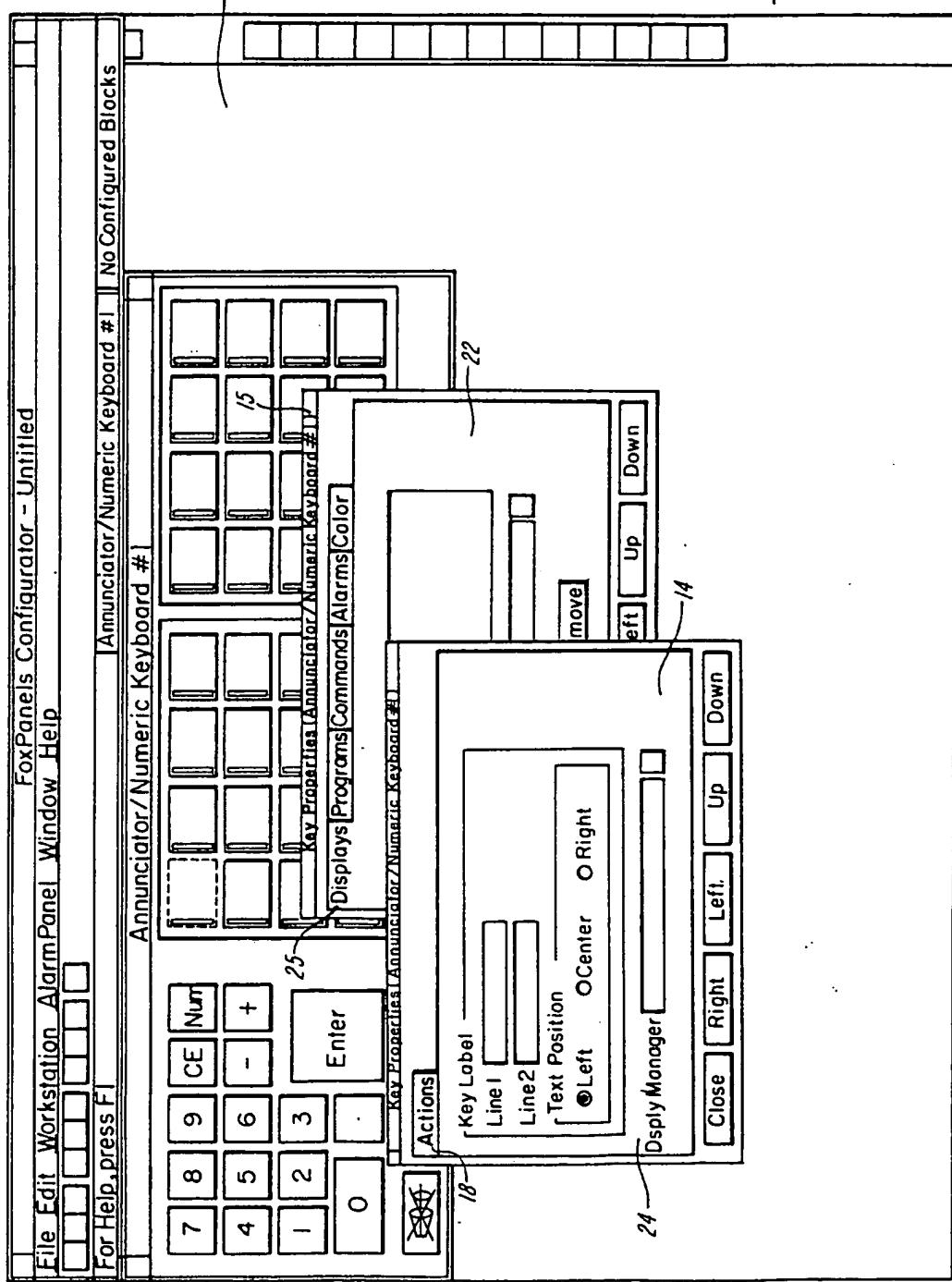
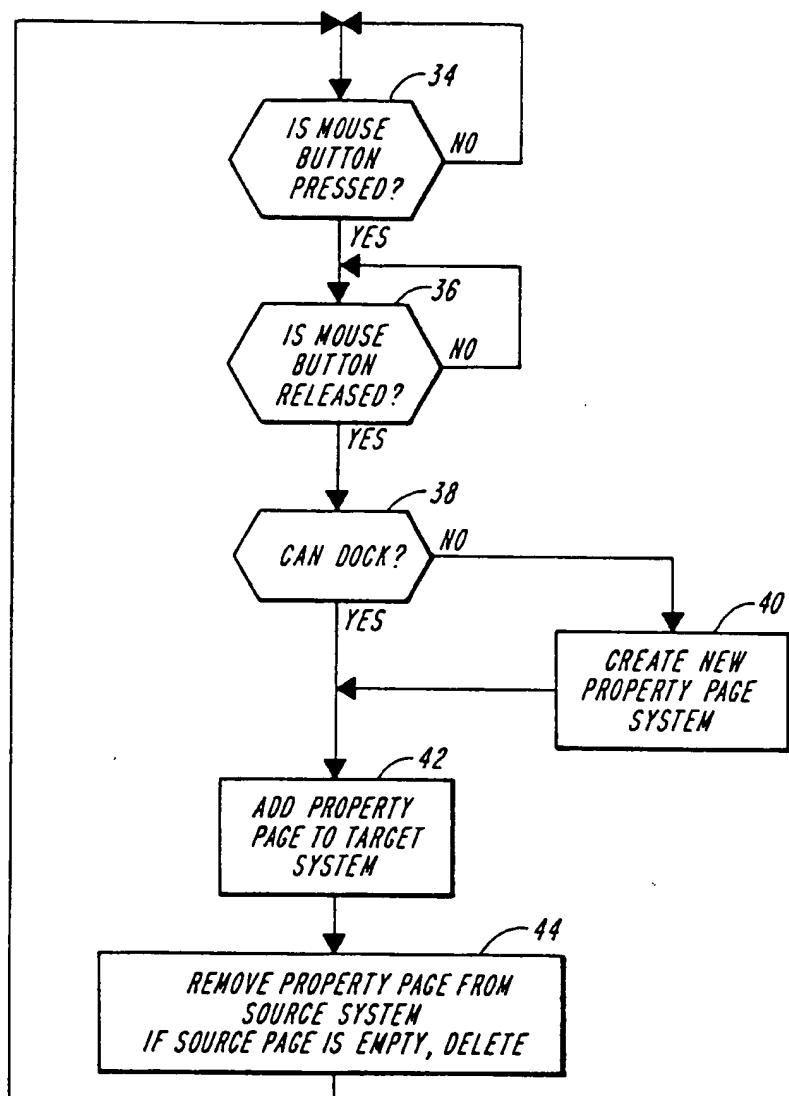
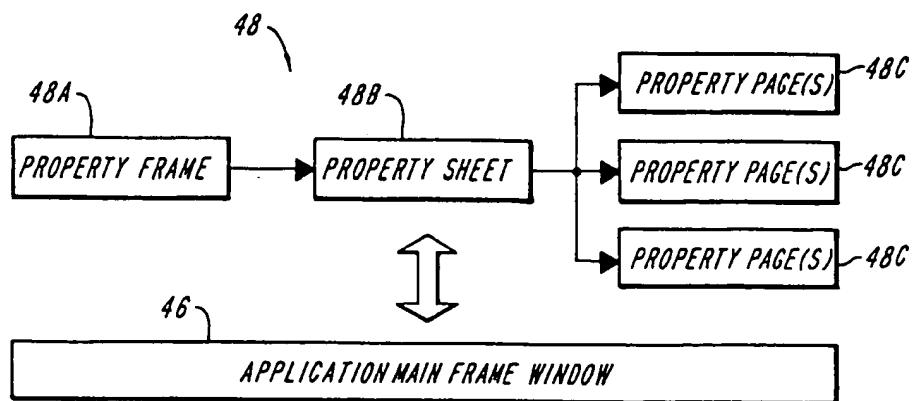


FIG. 4



**FIG. 6****FIG. 7**

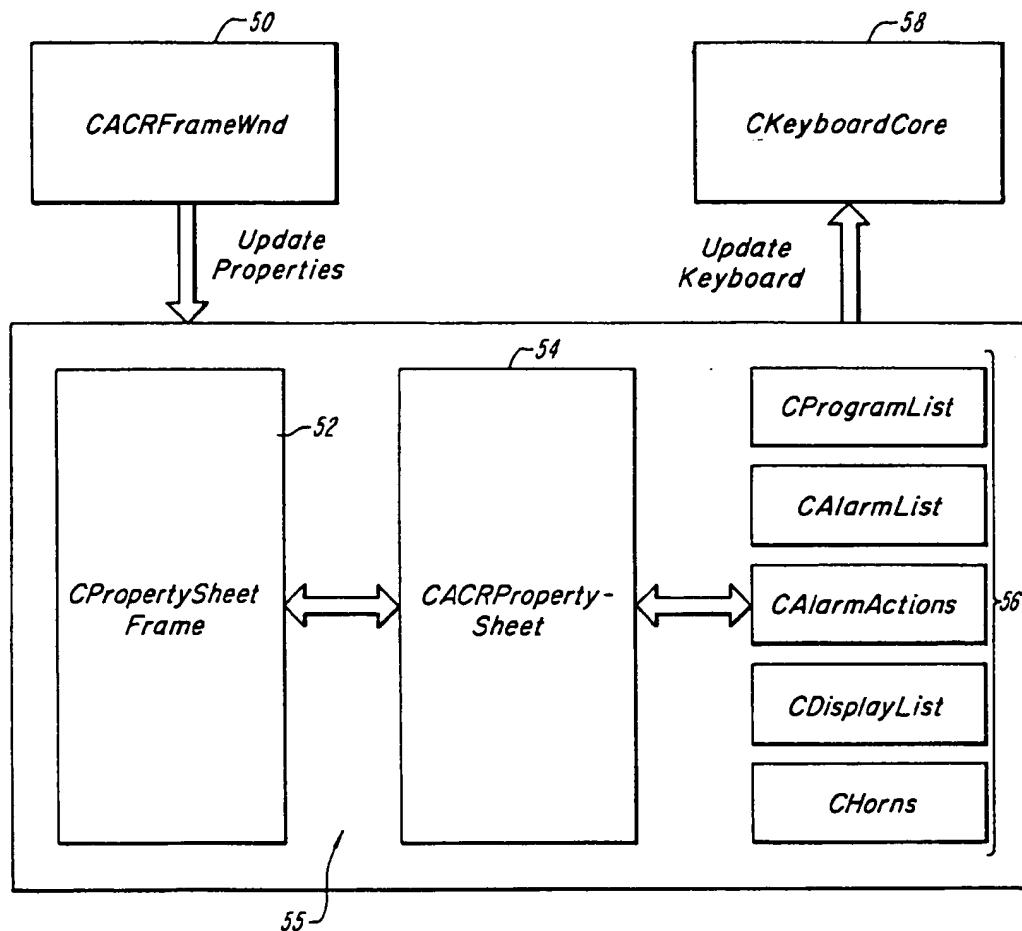


FIG. 8

